44 Pages

150 S.P

الأسبوع الأحمر

برمجة 1

- مقدمة
- حلقات الشرط
- حلقات التكرار
- مصفوفات
- التوابع
- تمارين محلولة – دورات

هندسة الميكانيك العام

السنة الثانية

#VivaRBCs

فريق الكريات الحمراء التطوعي

RBCs HAMAK

مقدمة برنامج C++

طريقة كتابة برنامج بلغة الـ C++

مثال:

- سنكتب البرنامج كاملاً ثم نشرح كل سطر على حدة.

```
(1)  #include < iostream.h >
(2)  int main ( )
(3)  {
(4)  int x1, x2, y;
(5)  cout << "Enter the first No" << endl;
(6)  cin >> x1;
(7)  cout << "Enter the second No" << endl;
(8)  cin >> x2;
(9)  y = x1 + x2;
(10) cout << "The result is" << y << endl;
(11) return 0;
(12) }
```

الشرح

هناك مكتبات تحوي على التعليمات التي نحتاجها في البرمجة.

يتم تضمين هذه المكتبات في السطور الأولى من البرنامج

فالسطر(1):

 # include < iostream.h >

فهنا نقوم باستدعاء المكتبة (***input output stream***)، حيث ***h*** هو اختصار لـ ***header file*** أي ملف رأسي، وهنا يتم تخزين الملفات التي لها علاقة بالدخل والخرج.

السطر(2):

حيث ***int*** هي اختصار لـ ***integer***، و ***main*** هي الدالة الأساسية وهي ثابتة ودائما يتم تنفيذ التعليمات بدء من هذا التابع.

السطر الـ (3) والـ (12):

يحتويان على قوس { الـ ***begin*** وقوس الـ } ***end*** وداخل هذان القوسان نضع جميع التعليمات.

كل قوس ***begin*** يجب وجود قوس ***end*** له.

السطر الـ (4):

تحتاج إلى تعريف الأرقام التي نريد إدخالها والتي نريد الحصول عليها فالمتغيرات y, x_1, x_2

تحجز أماكن للأعداد التي سنحتاجها فمثلاً عند إدخال العدد الأول فإنه يحل محل x_1 وهكذا......

السطر الـ (5):

بما أن البرنامج يتضمن إدخال الرقم الأول فلا بد من ظهور رسالة تخزين بأن أدخل الرقم الأول وهذا يتم بالتعليمة $cout$:

$$cout << "Enter\ the\ first\ No." << endl;$$

الـ $<<$ تكون بؤرتا السهم للخارج.

أي شيء يكتب بين " " $(Double\ Citation)$ يظهر كما هو على الشاشة.

$endl$ هي اختصار لـ $end\ line$ أي الانتقال لسطر آخر أما إذا وضعنا end تعني إنهاء التعليمة ولكن تبقى ضمن نفس السطر.

السطر الـ (6):

لإدخال الرقم الأول نستخدم تعليمة هي cin:

$$cin >> x;$$

الـ $>>$ بؤرتا السهم للداخل.

نضع الفاصلة المنقوطة لإنهاء التعليمة.

السطر الـ (9):

عملية جمع بسيطة $y = x_1 + x_2;$

السطر الـ (10):

لإظهار العدد الذي نريده نستخدم تعليمة $cout$

$$cout << "The\ result\ is" << y << endl;$$

نلاحظ أن: $<< y$

هي التي تظهر قيمة العدد الذي يعبر عن المتغير y فلو وضعنا

$$cout << "The\ result\ is\ y" << endl;$$

فسيظهر على الشاشة حرف الـ y وليس بنتيجة الجمع لأنه كما قلنا سابقاً أي شيء يكون بين " " يظهر كما هو على الشاشة.

وأخيراً لإنهاء البرنامج نعيد القيمة التي تريدها كما في السطر الـ (11)

$$return 0;$$

أنماط تعريف المتغيّرات في C++

للمحارف:

$Char$: من كلمة $Character$ حجمه $1\ Byte$ يُخزن فيه محارف مثل: $'a','b','c'$

يأخذ قيم صحيحة من $127 \Rightarrow -128$ فيمكن تخزين أرقام فيه.

ملاحظة: يجب وضع إشارة ' ' لتحديد المحرف.

للأعداد الصحيحة:

$Short$ حجمه $1\ Byte$

$integer$ حجمه $4\ Byte$

$long$ حجمه $8\ Byte$

للأعداد الحقيقية:

نفس الأعداد لكن مع فواصل عشرية:

$float$ حجمه $4\ Byte$

$double$ حجمه $8\ Byte$

$Long\ double$ حجمه $10\ Byte$

تعليمة الشرط if

أولاً سنضع البرنامج ثم نشرح الأسطر الجديدة:

```
#include < iostream.h >
void main( )
{
double x, y, F;     (1)
cout << "Input x, y" << endl;
cin >> x >> y; //cin >> x; cin >> y;  (2)
```

```
if (x < 0)   (3)
{
if (x > y)   (4)
F = x/(x − y);
else
F = y/(x − y);
else
```

```
if (x < y)
{
F = x * x;
}
else
{
F = y * y;
}
```

```
cout << "x = " << x << ", y = " << y << ", F = " << F << endl;    (5)
}
```

ملاحظات

(1):

عند عدم تحديد نوع المتغيرات يفضل استخدام $double$ لتعريف المتغيرات، لأنها لتعريف الأعداد الحقيقية.

ولا يشترط تعريف المتغيرات بعد قوس الـ $begin$ مباشرة ولكن المهم أن يكون قبل إدخال المتغيرات أي قبل الـ cin

ملاحظة: يستطيع المبرمج تسمية المتغير بما يريد ولكن يفضل أن يسميه برموز تدل على المتغير حتى يسهل على المستثمر قراءة البرنامج.

مثل: Avg لـ $average$ أي المتوسط.

Sum للجمع.

ولكن يشترط أن يبدأ المتغير بمحرف وليس عدد، فلا نستطيع متغير $3x$ لأنه يبدأ بعدد.

ولا نستطيع تعريف متغير بكلمات محجوزة مثل $cout$ وcin وغيرها من التعليمات.

(2):

أيضاً قمنا بوضع إشارة // لوضع تعليق نقصد منه أنَّ عملية الإدخال يمكن أن تتم بالشكل:

$$cin >> x >> y;$$

أو الشكل:

$$cin >> x;$$
$$cin >> y;$$

(3):

التعليمة if: تستخدم للشروط وفحصها وتنفيذ العمليات فيما إذا كان الشرط صحيح أو لا.

في الخوارزمية السابقة نقوم باختبار الشرط $x < 0$

فإذا كان صحيح ننفذ عمليات معينة وإذا كان خطأ ننفذ عمليات أخرى.

فإذا لم يتحقق الشرط نضع $else$ وننفذ العملية.

ولا يشترط أن يكون لكل تعليمة if تعليمة $else$ خلفها.

فهناك برامج تتطلب تنفيذ العمليات إذا كان الشرط صحيح فقط.

وتكتب التعليمة if على الشكل التالي:

```
if (الشرط) إذا
{
تنفيذ العمليات في حال كان الشرط صحيح
}
else وإلا
{
تنفيذ العمليات في حال كان الشرط خاطئ
}
```

:(3)

الخوارزمية تتطلب القيام بعملية معينة إذا كانت $x < 0$

هذه العملية هي اختبار إذا كان $y > x$ أم لا

لذلك قمنا باختبار هذا الشرط بوضع الدالة if

$$if(x > y)$$

ملاحظة: إذا كان الشرط وحيد ولا يحتوي بداخله على شروط أخرى فلا داعي للأقواس { }

لذلك نلاحظ أننا لم نضع قوس } بعد if في الســطر (4) لأن الدالة if لم تحتوي على شروط أخرى.

في باقي السطور قمنا بإكمال العمليات المطلوبة في الخوارزمية إذ قمنا باختيار الشرط $y > x$ وتنفيذ العمليات في حال كان هذا الشرط صحيح أولا ثم قمنا بتنفيذ العمليات المطلوبة في حال لم يتحقق الشرط $x < 0$ في السطر (3) حيث قمنا باختيار الشرط $x < y$ وتنفيذ العمليات في حال كان الشرط صحيح أم لا.

ملاحظة: العمليات الحسابية في البرمجة:

+ الجمع	− الطرح
/ القسمة	* الضرب

:(5)

أخيراً قمنا بإظهار القيم التي أدخلناها x, y والقيمة التي حسبناها F.

حيث:

$$cout << "x = " << x << ", y = " << y \ldots\ldots$$

تعني أن يظهر على الشاشة النص $x =$ ثم تظهر بعده قيمة x أي القيمة 3 ثم في نفس السطر يظهر النص $, y =$ ثم قيمة y أي 5 وهكذا ....

لو وضعنا :

$$cout << "x = " << x << endl;$$
$$cout << ", y = " << y$$

لظهر $x = 3$

$, y = 5$

أي لا تظهر على نفس السطر بسبب وجود $endl$

الإخراج على الشاشة: حسب قيم الخوارزمية السابقة: أي أنَّ:

$x = 3, y = 5, F = 9$

$Input\ x, y$

اكتب المعادلة هنايقوم المستخدم بإدخال x, y.

$x = 3, y = 5, F = 9$

تعليمة الـ switch

تستخدم هذه البنية جملة الشرط المغلقة *Close Condition* أي الشرط الذي يأخذ قيمة محددة (رقمية أو محرفية) ولهذا يمكن القول بأن هذه البنية ممكن أن تتعامل مع متحول واحد ممكن أن يأخذ في كل مرة قيمة محددة من خلال مجموعة القيم.

اكتب برنامج بلغة الـ $C++$ تقوم من خلاله بـ :

1– إدخال رقم الشهر وطباعة اسم الشهر الموافق للرقم.

2– إدخال رقم الشهر وطباعة اسم الفصل الموافق وذلك باستخدام *switch*.

(البرنامج):

```
#include < iostream.h >
 void main( )
 {
 int n;          (1)
 cout << "Enter n" << endl;       (2)
 cin >> n;   (3)
 switch(n)     (4)
 {
 case 1:    (5)
 cout << "january" << endl;
 break;
 case 2:
 cout << "february" << endl;
 break;
 case 3:
 cout << "march" << endl;
 break;
 case 4:
 cout << "April" << endl;
 break;
 .
 .
 .
 .
 case 12:
 cout << "december" << endl;
 break;
 default;      (6)
 cout << "Error" << endl;
 break;
 }
 }
```

شرح البرنامج

في السطر (1)و(2) و(3):

قمنا بتعريف متغير لرقم الشهر n وإظهار رسالة للمستخدم ليدخل رقم الشهر ثم إدخال رقم الشهر.

هيكلية $switch$

$switch$ (المتغير الذي نريد اختباره)

{

$case$ (الحالة الأولى) :

عمليات برمجية

$break;$ (لكسر البنية في حال تحققت الحالة الأولى)

:

$default;$ (في حال تم إدخال حالة غير موافقة لجميع الحالات التي وضعناها في البرنامج)

عمليات برمجية

}

السطر (4):

قمنا بوضع تعليمة $Switch$ لاختبار رقم الشهر n

السطر (5):

وضعنا الحالة الأولى وهي إدخال الشهر الذي رقمه 1

فيعطينا البرنامج أن اسم الشهر هو *january* وهكذا....

السطر(6):

في حال أدخلنا رقم ليس من 1 إلى 12 يظهر البرنامج لنا رسالة *Error*.

البرنامج للطلب الثاني:

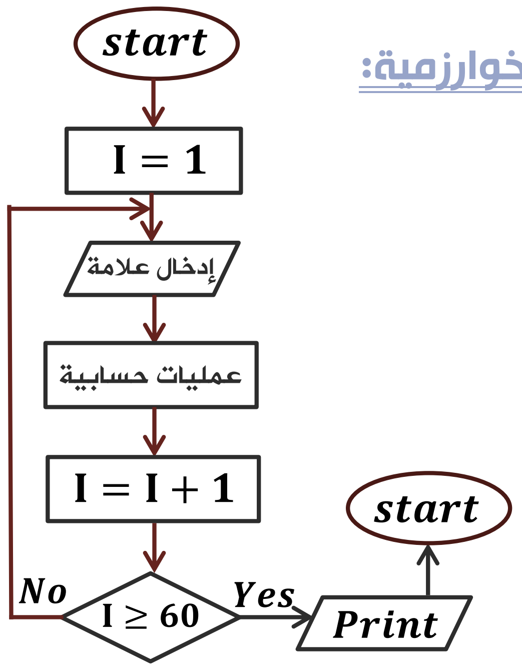
```
#include < iostream.h >
using namespace std;
void main( ) {
int n;
cout << "Enter n" << endl;
cin >> n;
switch(n)
{
case 12:     (1)
case 1:      (1)
case 2:      (1)
cout << "winter" << endl;
break;
case 3:
case 4:
case 5:
cout << "spring" << endl;
break;
case 6:
case 7:
case 8:
cout << "summer" << endl;
break;
case 9:
case 10:
case 11:
cout << "Autumn" << endl;
break;
default;
cout << "Error" << endl;
}
```

حلقات التكرار

أنواع حلقات التكرار:

1. حلقة *for*.
2. حلقة *while*.
3. حلقة *do while*.

الخوارزمية:



اكتب برنامج بلغة الــ ++C يقوم المستخدم من خلاله بإدخال علامات الطلاب في مادة البرمجة وحساب العلامة الدنيا والعلامة العظمى ومتوسط العلامات علماً أنَّ عدد الطلاب 60.

شرح الخوارزمية:

عندما يطلب البرنامج تكرار عملية معينة لأكثر من مرة كالبرنامج السابق) حيث طلب حساب العلامة الدنيا والعظمى والمتوسط لــ 60 طالب) ولا نستطيع كتابة التعليمة 60 مرة لذلك هناك ما يسمى بحلقات التكرار حيث أننا نكتب العملية مرة واحدة ضمن حلقة ونحدد عدد المرات التي نريد أن نكرر فيها العملية.

في الخوارزمية السابقة قمنا بإعطاء قيمة للعداد هي $I = 1$ وبعد إدخال العلامة والقيام بالعمليات الحسابية ثم قمنا بزيادة العدّاد بمقدار 1: $I = I + 1$

ثم وضعنا:

$$I \geq 60$$

أي إذا كانت قيمة العداد مساوية أو أكبر من 60 نقوم بإنهاء البرنامج لأننا بذلك نكون قد حسبنا المتوسط والعلامة العليا والدنيا لــ 60 طالب، أما إذا كان العدّاد أقل من 60 نكرر العملية.

البرنامج:

```
#include < iostream.h >
void main ( )
{
int mrk, min, max, sum;
double avg;
min = 100;
max = 0;
sum = 0;
for ( int i = 1; i <= 60; i + +)                (1)
{
cout << "Input the mark of the student" << i << endl;        (2)
cin >> mrk;
if(mrk > max)
max = mrk;
if(mrk < min)
min = mrk;
sum = sum + mrk;//sum+= mrk;                (3)
}
avg = sum/60;                (4)
cout << "min = " << min << endl;
cout << "max = " << max << endl;
cout << "Avg = " << avg << endl;
}
```

شرح البرنامج

السطر (1):

الحلقة for تقسم إلى ثلاثة أقسام:

$for(1\ ;\ 2\ ;\ 3\ )$

القسم الأول: تهيئة، حيث نقوم بإسناد قيمة للعدّاد حيث قمنا بالبرنامج السابق بتعريف متغير اسمه i ونسند له قيمة $i=1$.

القسم الثاني: تعيين شرط الإنهاء أي عدد مرات تكرار العمليّة.

القسم الثالث: زيادة العدّاد.

في البرنامج السابق نقصد من السطر (1) التالي:

من أجل $i=1$ وحتى تصبح 60 نفّذ العمليات التالية وذلك مع زيادة العدّاد وعندما تصبح أكثر من 60 ننهي الحلقة.

السطر (2):

```
cout << "Input the mark of the student" << i << endl;
```

نلاحظ وضع القيمة i بعد النص الذي يطلب من المستخدم إدخال علامة الطالب.

وبما أنَّ i تزداد كل مرة بمقدار 1 لذلك فكل مرة تتغير الرسالة للمستخدم أي يظهر:

أدخل علامة الطالب 1

أدخل علامة الطالب 2

أدخل علامة الطالب 3

وهكذا....

ملاحظة: لم نضع الـ i ضمن علامتي التنصيص لأن كل شيء ضمن هاتين العلامتين " " يظهر كما هو على الشاشة.

مناقشة حالة الـ min والـ max

نسند للـ min أعلى قيمة 100 وللـ max أدنى قيمة 0

السبب:

بمقارنة العلامات:

عند إدخال علامة أول طالب ولتكن 70

نختبر العلامة $if(mrk > max)$

ونلاحظ بأنَّ $70 > 0$ فإن $max = 70$

ثم نختبرها $if(mrk < min)$

نلاحظ أنَّ $70 < 100$ فإنَّ $min = 70$

فعند أول طالب يكون $min = max$.

ندخل علامة ثاني طالب ولتكن 79، نختبر العلامة $if(mrk > max)$ ونلاحظ أنَّ $(79 > 70)$ صحيح فإن $max = 79$

ونختبرها $if(mrk < min)$ نلاحظ أنَّ $(79 < 70)$ خطأ لذلك تبقى $min = 70$

وهكذا نكمل العملية لــ 60 طالب ونستطيع معرفة أدنى وأعلى علامة.

السطر (3):

نقوم بحساب المجموع لاستخدامه في المتوسط الحسابي.

مجموع علامات الطلاب: $sum = sum + mrk$

قمنا بإسناد قيمة الصفر للمجموع $sum = 0$ حتى لا يبدأ من قيمة عشوائية.

فعند إدخال علامة الطالب الأول ولتكن 70 ستكون:

$$sum = 0 + 70 \ \rightarrow \ sum = 70$$

فأصبحت sum تأخذ القيمة 70 ثم ندخل علامة الطالب الثاني:

$$sum = 70 + 79 \ \rightarrow \ sum = 149$$

فأصبحت sum تأخذ قيمة 149 وهكذا حتى نكمل العملية لــ 60 طالب.

السطر (4):

نقوم بحساب المتوسط الحسابي لعلامات الطلاب وهو تقسيم المجموع على عدد الطلاب وهو 60.

ملاحظة: قمنا بوضع المتوسط خارج الحلقة لأننا نريد حساب متوسط واحد لمجموع علامات كافة الطلاب.

حلقة $while - do\ while$

تعليمة While:	تعليمة do while:

هيكلية التعليمة:
$while\ (i <= 100)$
{
عمليات برمجة
$i++;$//الزيادة
}
زيادة العداد تكون في آخر سطر ضمن قوسين العمليات.
	هيكلية التعليمة:
	do
	{
	عمليات برمجة
	$i++;$//الزيادة
	}
	$while\ (i <= 100)\ ;$

الفرق بين تعليمة $while$ وتعليمة الـ $do\ while$:

في تعليمة الــ $while$ نقوم باختبار الشرط ثم تنفيذ التعليمات أما في تعليمة الــ $do\ while$ يقوم بتنفيذ التعليمة ثم اختبار الشرط أي أنه في حالة الــ $do\ while$ يكون مجبر على الدخول بالحلقة مرة واحدة على الأقل.

في تعليمة الــ $while$ لا نضع فاصلة منقوطة (;)

$$while\ (i <= 100)$$

أما في تعليمة الــ $do\ while$ نضع فاصلة منقوطة (;)

$$while(i <= 100)\ ;$$

مثال: اكتب برنامج بلغة C++ يقوم المستخدم من خلاله بإدخال علامات الطلاب في مادة البرمجة وحساب العلامة العظمى والعلامة الدنيا ومتوسط العلامات علماً أنَّ عدد الطلاب غير محدد يدخله المستخدم.

الحل:

- باستخدام حلقة while:

```
#include < iostream.h >
void main ()
{
int mrk, min, max, sum, n;
double avg;
min = 100; max = 0; sum = 0;
cout << "Input N" << endl;    (1)
cin >> n;
int i = 1;   (2)
while( i <= n)      (3)
{
cout << Input the mark No << i << endl;
cin >> mrk;
if(mrk < min)
 min = mrk;
if(mrk > max)
max = mrk;
sum+= mrk;//sum = sum + mrk;
i + +;//i = i + 1;
}
Avg = sum/ n;
cout << "min = " << min << ", max = " << max << ", avg = " << Avg << endl;
}
```

حيث:

mrk : هي علامة الطالب , min: هي اقل علامة

max: اعلى علامة , sum: مجموع العلامات

avg: متوسط الحسابي للعلامات , n: عدد الطلاب

شرح البرنامج:

السطر (1):

يحتوي السطر (1) على إعلان للمستخدم لكي يدخل عدد الطلاب.

السطر(2) :

أعطينا قيمة ابتدائية للعداد وهي $i = 1$، ثم وضعنا تعليمة $while(i <= n)$ في السطر (3) أي أننا نقصد بالتعليمة التالي: طالما $i <= n$ أصغر أو تساوي الـ n (عدد الطلاب الذي أدخله المستخدم) قم بتنفيذ العمليات.

البرنامج السابق باستخدام تعليمة ***do while***:

```
#include < iostream.h >
void main ( )
{
int n, mrk, min, max, sum;
double avg;
cout << "Input N" << endl;
cin >> n;
int i = 1;
min = 100; max = 0, sum = 0;
do
{
cout << "Input mark No" << i << endl;
cin >> mrk;
if(mrk < min)
min = mrk;
if(mrk > max)
max = mrk;
sum+= mrk;
i + +;
}
while(i <=  n);
Avg = sum/ n;
cout << "min = " << min << ",max = " << max << ", avg = " << Avg << endl;
}
```

ملاحظة:

```
sum *= 5;//sum = sum * 5;
sum/= 5;//sum = sum/ 5;
```

حلقات for المتداخلة

عندما نحتاج في البرنامج إلى عدادين أو أكثر أو يكون لدينا عمليتين مستقلتين أو أكثر تحتاج إلى تكرار، نستخدم حلقات for المتداخلة، ففي البرنامج السابق استخدمنا حلقة for واحدة لأنه كان لدينا عملية واحدة فقط تحتاج إلى تكرار.

الهيكلية:

```
for (int i = 1; i ≤ 100; i + +)
    for(int j = 1; j ≤ 500; j + +)
    {
        التعليمات البرمجية
    }
```

مثال: في شركة تجارية مؤلفة من 50 موظف ولكل موظف 5 مهام، يتقاضى الموظف راتب مقطوع مقداره 20000 ويتقاضى مبلغ إضافي مقداره 10% عن المبلغ المقطوع لكل مهمة ينجزها.

اكتب برنامج بلغة $C++$ نقوم من خلاله بحساب دخل كل موظف في الشركة.

البرنامج:

```
#include < iostream.h >
int main( )
{
double s[50];     (1)
double inc;           (2)
for ( int i = 0; i < 50; i + +)
s[i] = 20000;

for(int i = 0; i < 50; i + +)
{
    for(int j = 1; j ≤ 5; j + +)
{
cout << "Input task No" << j << endl;      (3)
cin >> inc;
s[i] = inc ∗ 0.1 + s[i];
}
cout << "s[<< i >>] = " << s[i] << endl;
}
return 0;
}
```

المثال السابق هو عبارة عن مثال استخدمنا فيه حلقات ***for*** المتداخلة كما استخدمنا مصفوفة أحادية البعد.

حيث أن المصفوفة أحادية البعد هي نوع من أنواع بنى المعطيات، لها عدد محدود ومرتب من العناصر التي تكون جميعها من نفس النوع، فمثلاً يمكن أن تكون جميعها صحيحة ***int*** أو حقيقية ***double*** أو محرفية ***char*** ولكن لا يمكن الجمع بين نوعين مختلفين من نفس المصفوفة.

ففي المثال السابق استخدمنا المصفوفة $s[50]$ وهي مصفوفة اسمها s عدد عناصرها **50**.

ملاحظة:

استخدمنا المصفوفة في المثال السابق حتى تخزن راتب كل موظف في عنصر من عناصر المصفوفة لذلك اخترنا أن يكون عدد عناصرها **50** عنصر حتى لا تختلط رواتب الموظفين مع بعضها عند استخدام المصفوفات أولاً نقوم بتعريف المصفوفة من النمط الذي نريده.

شرح البرنامج:

السطر(1): قمنا بتعريف المصفوفة من النوع double

السطر (2): قمنا بتعريف متغير من النمط inc من كلمة $incoming$ هذا المتغير يدخله المستخدم يعبّر عن المبلغ الذي استحصله الموظف من المهام.

في السطر (3) والسطر (4):

قمنا بتهيئة المصفوفة بالرقم 20000 لأن راتب الموظف الثابت 20000 والزيادة تكون فوق الراتب الأصلي.

ملاحظة: تهيئة المصفوفة تتم بحلقة for مستقلة.

أي أننا وضعنا في المثال السابق حلقة for لتهيئة المصفوفة ثم قمنا بوضع حلقتي for الحلقة الأولى للموظفين (الـ 50 موظف).

والحلقة الثانية لمهام كل موظف (الـ 5 مهام).

تعليمة *break*

```
for ( int i = 1; i <= 10; i ++)
{
if ( i == 5)
break;
cout << i << endl;
}
```

نلاحظ وضع ;*break* بعد $if(i == 5)$

أي تعني أنه عندما تصبح $i = 5$ يتم كسر الحلقة وإنهاء البرنامج فيكون الخرج:

```
1
2
3
4
Press any key to continue ...
```

break

تستخدم هذه التعليمة مع البنى التكرارية وتسبب عند تنفيذها الخروج نهائيا من البنية والانتقال الى اول تعليمة تليها مباشرة

تعليمة continue

```
for(int i = 1; i <= 10; i++)
{
if(i == 5)
continue;
cout << i << endl;
}
```

نلاحظ وضع ;continue بعد $if(i == 5)$

أي تعني أنه عندما $i = 5$ نقوم بتخطي هذه التعليمة ثم الإكمال مع التعليمات التي بعدها فيكون الخرج كالتالي:

```
1
2
3
4
6
7
8
9
10
Press any key to continue ...
```

continue

تستخدم هذه التعليمة مع البنى التكرارية فقط وتسبب تجاوز ما تبقى من التعليمات في جسم الحلقة والانتقال الى تكرار جديد

متواليات الهروب

1 – $\backslash n$: تعني الانتقال لسطر جديد وتكافئ ;endl

```
cout << "student is \n";
cout << "clever";
```

الخرج:

student is clever

$\backslash n \backslash n \backslash n$: تعني ترك ثلاثة أسطر.

---

2 – $\backslash t$: تترك فراغ يخزن فيه 8 بت.

```
cout << "student\t is";
```

الخرج:

student   is

---

3 – \\\\: لطبع إشارة \\

```
cout << "student\\is";
```

الخرج:

student \is

---

4 – ?\\: لطبع إشارة استفهام.

```
cout << "student \? is";
```

الخرج:

student? is

---

5 – "\\: لطبع الإشارة "

```
cout << "student\" is";
```

الخرج:

student " is

---

6 – $\backslash r$: تعليمة إجبار المؤشر على الانتقال لبداية السطر.

```
cout << "student\r";
cout << "clever";
```

الخرج:

cleverstudent

7– $\backslash b$: يرجع المؤشر إلى الوراء بمقدار خطوة.

$$cout \ll "student\backslash b";$$

الخرج:

clever

يعود المؤشر إلى ما قبل حرف الـ r.

8– $\backslash a$: يعطي طنين.

يستفاد منه لمعرفة ما إذا تم إنهاء مرحلة أم لا، أي عندما نضع عند مرحلة ما \a فعندما يمر البرنامج على هذه المرحلة يصدر طنين، أي يستخدم لتنقيح كود البرمجة.

ملاحظات على تعليمة cout:

- ✓ نضع بعدها الإشارة >>.
- ✓ النص الذي نريد أن نظهره نضعه ضمن علامتي تنصيص " ".
- ✓ تستخدم تعليمات الهروب ضمن النص فقط أي ضمن إشارتي التنصيص " ".
- ✓ لإخراج قيمة متحول ما نضع اسم المتحول ضمن تعليمة الاخراج دون علامتي تنصيص.

الفرق بين i++,++i

$i++$: يتم التنفيذ قبل الجمع.

مثال:

$$i = 3;$$
$$cout \ll i++ \ll endl;$$

فإن القمة التي سيعطيها في الخرج التنفيذي 3 وليس 4 أي أنّ القيمة الأولى التي يعطيها تكون قبل الزيادة.

أما:

$++i$: يتم الجمع قبل التنفيذ.

مثال:

$$i = 3;$$
$$cout \ll ++i \ll endl;$$

فإن القمة التي سيعطيها في الخرج التنفيذي هي 4 لأنه جمع، أي أنّ القيمة التي يعطيها تكون بعد الزيادة.

أما في حال:

$i = 3;$ $i++;$ $cout \ll i \ll endl;$	$i = 3;$ $++i;$ $cout \ll i \ll endl;$

سيكون الخرج 4 في الحالتين لأنه جمع قبل الطباعة.

وبنفس الأسلوب $--i$ و $i--$.

المصفوفات

أنواع المصفوفات

1– مصفوفة أحادية البعد: كل عنصر له حدث واحد x.

2– متعددة الأبعاد: ثنائية الأبعاد... ثلاثية الأبعاد...إلخ

تعريف المصفوفة:

نضع أولاً نمط المعطيات $(double, int, ..)$ ثم اسم المصفوفة ثم أقواس مربعة ونكتب ضمن القوسين عدد خانات المصفوفة وفاصلة منقوطة.

مثال على مصفوفة أحادية البعد:

$$double\ a[10];$$
$$int\ b[7];$$

قمنا بتعريف المصفوفة التي اسمها a ووضعنا نمط معطياتها من النمط $double$ ثم وضعنا ضمن القوسين عدد خانات (عناصر) المصفوفة التي هي 10 عناصر.

أيضاً قمنا بتعريف مصفوفة اسمها b ووضعنا نمط معطياتها من النمط int ثم وضعنا ضمن القوسين عدد خانات (عناصر) المصفوفة التي هي 7 عناصر.

شروط القيمة التي تحتويها المصفوفة:

أن تكون عدد ثابت

أي لا يجوز وضع $int\ a[n]$ حيث أن n متغير

حتى لو عرفنا n وأدخلناها أي:

$$int\ n;$$
$$cin \gg n;$$
$$int\ a[n];$$

هذه الطريقة أيضاً خاطئة.

أما عندما نعرّف n على أنه ثابت أي:

$$const\ int\ n = 7;$$
$$int\ a[n];$$

هذه الكتابة صحيحة لأن الـ n عرّفناه على أنه ثابت.

> **ملاحظة**: عناصر المصفوفة تبدأ من الصفر أي إذا كان عدد عناصر المصفوفة 14 $(s[14])$ فتكون القيم من 0 حتى 13.

في عملية إسناد قيم لعناصر المصفوفة وليكن عدد عناصرها 10:

$$a[1] = 13;$$
$$a[9] = 27;$$
$$a[10] = 100; Error$$

أظهر رسالة $Error$ لأن عدد العناصر للمصفوفة هي 10 عناصر وتبدأ من الصفر فتنتهي بالـ 9.

مثال:

اكتب برنامج بلغة $C++$ تقوم من خلاله بحساب معدل تخرج طالب هندسة الميكانيك العام و إذا علمت أن عدد السنوات التي يقضيها في الجامعة 5 سنوات وأن لديه في كل عام جامعي 14 مادة على أن يتم إدخال علامات الطالب من قبل المستخدم لكل عام على حدى مستخدماً مصفوفة أحادية البعد لتخزين علامات الطالب في كل سنة.

البرنامج:

```
#include < iostream.h >
void main ()
{
int n, s;     }   (1)
int a[14];    }
double Avg = 0;
for (int i = 1; i <= 5; i ++)      (2)
{
cout << "Enter the markes of the year No" << i << endl;
s = 0;
for(int j = 0; j < 14; j ++)              (3)
{
cout << "Input mark No: " << j + 1 << endl;     (4)
cin >> n;
a[j] = n;            (5)
s = s + a[j];        (6)
Avg+= s /14;         (7)
}
}
Avg = Avg/5;      (8)
cout << "Average = " << Avg << endl;
}
```

شرح البرنامج

السطر (1):

قمنا بتعريف المتغير n لعلامة الطالب، ثم قمنا بتعريف مصفوفة اسمها a وعدد عناصرها 14 عنصر.

السطر (2):

وضعنا حلقة for الأولى للخمس سنين ووضعنا ضمنها إعلان $cout$ لإدخال علامات السنة i.

السطر (3):

وضعنا حلقة for ثانية للـ 14 مادة

السطر (4):

إعلان للمستخدم لإدخال العلامة $1 + j$، لم نسند القيمة j لأن أول قيمة لـ j في حلقة for كانت صفر فحتى لا يظهر للمستخدم: أدخل العلامة 0 أسندنا القيمة $1 + j$.

السطر (5):

بعد إدخال العلامة لنقوم بإسنادها للمصفوفة أي تخزينها في أحد عناصر المصفوفة فمثلاً إذا كانت أول علامة أدخلها المستخدم 88 سيكون الإسناد كالتالي:

$$a[0] = 88$$

أي أول عنصر من عناصر المصفوفة خزنا فيه القيمة 88.

السطر (6):

قمنا بجمع علامات الطالب لحساب المتوسط الحسابي.

السطر (7):

قمنا بحساب المتوسط الحسابي وهو مجموع علامات المواد التي عددها (14مادة) ثم جمع جميع المتوسطات الحسابية للخمس سنوات:

$$Avg = Avg + s/14$$

حيث أننا أسندنا أول قيمة لـ Avg صفر $(Avg = 0)$

السطر (8):

قمنا بحساب المتوسط الكلي الذي يطلبه البرنامج وهو مجموع متوسط العلامات في جميع السنين على عدد السنوات (5 سنوات).

> **ملاحظة:** في البرنامج السابق لم نستفيد من المصفوفة في البرنامج ولكننا تعلمنا كيفية تنفيذ المصفوفة.

مثال من نوطة العملي:

اكتب برنامج بلغة $C++$ يطلب من المستخدم إدخال قيم عناصر مصفوفة أعداد حقيقية أحادية حجمها (25) عنصرثم يقوم بترتيب عناصر هذه المصفوفة ترتيباً تصاعدياً ثم يطبعها على الشاشة بعد ترتيبها.

```
#include <iostream.h>
void main()
{
double Arr[25];
for(int k = 0; k < 25; k++)
{
cout << "Arr[" << K << "] = ";    السطر 1
 cin >> Arr[k];
}
double Temp;            السطر 2
for(int i = 0; i < 24; i++)
{
for(int j = 0; j < 24 - i; j++)
{
if(Arr[j] > Arr[j + 1])
{
Temp = Arr[j];
Arr[j] = Arr[j + 1];
Arr[j + 1] = Temp;
}
}
}
for(int v = 0; v < 25; v++)
cout << Arr[v] << ",";
cout << endl;
}
```

السطر(1):

رسالة مخاطبة للمستخدم لإدخال عناصر المصفوفة ضمن حلقة for حيث تظهر الرسالة على الشكل التالي

$Arr[0] =$ ثم يقوم المستخدم بإدخال العنصر الأول و هكذا.

السطر(2): عرّفنا متغير من النوع $double$ واسمه $Temp$ سنستخدمه لاحقاً.

الغرض من حلقتي for المتداخلتين:

حلقة for الأولى لعدد تكرار المقارنات التي سنجريها.

حلقة for الثانية من $j = 0$ إلى $i - 24$ و السبب:

أننا قمنا بمقارنة كل عدد في المصفوفة مع العدد الذي يليه أي أننا مثلاً:

قمنا بمقارنة العدد الذي ترتيبه 0 في المصفوفة مع العدد الذي ترتيبه 1 في المصفوفة

فإذا كان $Arr[0]$ أكبر من $Arr[1]$ نبدل بين العددين فيصبح العدد الأكبر ترتيبه 1 و ليس 0 ثم نقارن بين العدد الذي ترتيبه 1 و العدد الذي ترتيبه 2

فإذا كان أكبر نبدل بينهما فيصبح العدد الأكبر ترتيبه 2 و هكذا..

فنلاحظ بإكمالنا المقارنات أن أكبر عدد أصبح آخر عدد في المصفوفة لذلك عندما ندخل في حلقة for للمرة الثانية لا داعي لأن تقارن مع آخرعدد من المصفوفة لذلك يكون عدد المقارنات 23 و ليس 24 أي $24 - i$.

كيفية عملية التبديل بين عنصرين في المصفوفة

نستعين بمتغير حتى نبدل بين عنصرين أسميناه هنا $Temp$

ثم وضعنا التعليمات:

$$Temp = Arr[j];$$
$$Arr[j] = Arr[j + 1];$$
$$Arr[j + 1] = Temp;$$

أولاً قمنا بإسناد قيمة العدد الذي ترتيبه في المصفوفة (دليله) j إلى المتغير $Temp$ ثم أسندنا قيمة العدد الذي ترتيبه $j + 1$ إلى $Arr[j]$ ليصيح ترتيبه j

و أسندنا قيمة الـ $Temp$ إلى العدد الذي ترتيبه $j + 1$

توضيح مثال:

سوف نبدل بين عنصرين في المصفوفة:

$$A[0] = 5$$
$$A[1] = 3$$

نريد أن يصبح $A[0] = 3$

$A[1] = 5$

نقوم بالتعليمات:

$Temp = A[0];$ $A[0] = A[1];$ $A[1] = Temp;$	أي: $Temp = 5$ $A[0] = 3$ $A[1] = 5$

- سنشرح الآن بمثال لمن لم يفهم الشرح النظري

مثال توضيحي:

سنوضح بمصفوفة حجمها 6 بدلاً من 25 فتصبح حلقتين for:

$$for(int\ i = 0; i < 5; i++)$$
$$\{$$
$$for(int\ j = 0; j < 5 - i; j++)$$
$$\{$$
$$if(Arr[j] > Arr[j+1])$$
$$\{$$
$$Temp = Arr[j];$$
$$Arr[j] = Arr[j+1];$$
$$Arr[j+1] = Temp;$$
$$\}$$
$$\}$$
$$\}$$

سنفرض أن الأعداد التي أدخلها المستخدم هي:

$\{2, 12, 1, 9, 6, 0\}$

$j = 0$

إذا كانت $A[0] > A[1]$ نبدل ...

هنا 2 ليست أكبر من 12 فلا نبدل بين العنصرين.

$j = 1$ أي دخلنا في حلقة for الثانية مرة ثانية.

إذا كانت $A[1] > A[2]$ نبدل..

هنا $12 > 1$ فنبدل بين العنصرين فتصبح المصفوفة على الشكل التالي:

$\{2, 1, 12, 9, 6, 0\}$

$j = 2$ دخلنا في حلقة for الثانية مرة ثالثة.

إذا كانت $A[2] > A[3]$ نبدل...

هنا $12 > 9$ فنبدل بين العنصرين و تصبح المصفوفة على الشكل التالي:

$\{2, 1, 9, 12, 6, 0\}$

$j = 3$ دخلنا في حلقة for الثانية مرة رابعة.

إذا كانت $A[3] > A[4]$ نبدل...

هنا $12 > 6$ فنبدل بين العددين و تصبح المصفوفة على الشكل التالي:

$\{2, 1, 9, 6, 12, 0\}$

$j = 4$ دخلنا في حلقة for الثانية للمرة الخامسة.

إذا كانت $A[4] > A[5]$ نبدل.

هنا $12 > 0$ فنبدل بين العنصرين

$\{2, 1, 9, 6, 0, 12\}$

الآن أصبحت $j=5$ أي لا تحقق شرط حلقة الـ for و هو أن تكون $j<5-i$

أي $j<5-0$ فنكسر الحلقة و ندخل في حلقة for الأولى مرة ثانية.

$j=0$

إذا كانت $A[1] > A[0]$ نبدل

هنا $2>1$ فنبدل و تصبح المصفوفة:

$\{1,2,9,6,0,12\}$

$j=1$

إذا كانت $A[2] > A[1]$ نبدل

هنا 2 ليس أكبر من 9

$j=2$ إذا كانت $A[2] > A[1]$ نبدل

هنا $9>6$ نبدل و تصبح المصفوفة على الشكل التالي:

$\{1,2,6,9,0,12\}$

$j=3$

إذا كانت $A[4] > A[3]$ نبدل

هنا $9>0$ نبدل

$\{1,2,6,0,9,12\}$

أصبحت $j=4$ أي لا تحقق الشرط $j<5-i$ أي $j<5-1$ لذلك تكسر حلقة for الثانية و ندخل في حلقة for الاولى مرة ثالثة وهكذا ......

إلى أن نرتب جميع عناصر المصفوفة تصاعدياً

و هكذا .....

أضف ملاحظاتك

....

....

....

....

....

....

....

....

....

التوابع

<u>تعريفها</u> : هي عبارة عن اجراء يقوم بأداء مهمة أو مجموعة من المهام و يستدعى من قبل تابع اخر

مراحل استخدام التوابع :

- التصريح عن التابع : ويكتب بعد مرحلة ما قبل الترجمة
- استدعاء التابع : ويتم داخل جسم التابع الرئيسي $main()$ او ضمن جسم تابع اخر
- تعريف التابع : ويوضع خارج التابع الرئيسي $main()$

```
#include < iostream.h >          مرحلة ما قبل الترجمة.
int sum (int , int);     (1)      تصريح عن التابع.
void main()
{
statmantes;
cout << sum(a, b);      (2)       استدعاء التابع.
}
int sun (int x , int y)      (3)  تعريف التابع.
{
return x + y ;
}
```

ملاحظات

- تصريح التابع هو سطر واحد يبلغ المترجم عن اسم التابع وعدد وسطائه وانواعها ونوع القيمة المعادة بواسطة التابع.
- يتشابه تصريح التابع، السطر الأول في تعريف التابع لكن في سطر التصريح تلية فاصلة منقوطة وفي سطر التعريف نضع نوع وسطاء التابع وأسماء الوسطاء.
- تقسم التوابع الى قسمين: 1-توابع تعيد قيمه , 2-توابع لا تعيد قميه.
- التوابع التي تعيد قيمة: يتم التصريح عنها بـ $double$, int ولا بد من استدعائها ضمن تعليمة إخراج او اسناد ويتم اعادة القميه المطلوبة عن طريقة تعليمة الـ$return$ التي تكتب في تعريف التابع.

$$cout \ll sum(a,b);$$

- التوابع التي لا تعيد قيمة: يتم التصريح عنها بـ $void$ ويتم استدعائها بتعليمة مستقلة.

$$sum(a,b);$$

- يؤدي استدعاء التابع الى الانتقال الى مرحلة تعريف التابع.
- تابع العودية: هو التابع الذي يستطيع استدعاء نفسة اثناء تعريفة.

برمجة التوابع

إما أن تكون كاملة قبل الـ $main$ أونضع فقط الترويسة قبل الـ $main$.

سنقوم بحل البرنامج التالي بالطريقتين:

مثال:

اكتب برنامج بلغة $C++$ تقوم من خلاله بحساب قيمة عدد على أن يقوم المستخدم بإدخال أساس هذا العدد وقوته وذلك باستخدام التوابع وقم ببرمجة التابع ولا تستخدم التوابع الجاهزة من ملف $math.h$

نلاحظ أن النص طلب كتابة كود البرمجة دون استدعاء مكتبة الرياضيات وبالتالي لا نستطيع استخدام التابع $pow(x,y)$ وحل البرنامج مباشرةً

لذلك سنقوم ببرمجة تابع وسنسميه مثلاً: pwr ونبرمجه كما طلب النص.

البرنامج:

```
#include < iostream.h >
double pwr(double x, int y);     (1)
void main ()
{
double z, k;
int f;
cout << "Input z, f" << endl;
cin >> z >> f;
k = pwr(z, f);
cout << "k = " << k << endl;
}
double pwr(double x, int y)   (2)
{
double Res;
Res = 1;
for(int i = 1; i <= y; i ++)
Res = Res * x;
return Res;        (3)
}
```

شرح البرنامج

السطر (1):

قمنا بتعريف التابع من النمط $double$ وهو نمط المعطيات الذي يريد المبرمج من التابع أن يعيده.

ثم قمنا بتسمية التابع pwr وداخل التابع عرّفنا المتغير x من النمط $double$ للأساس والمتغير y من النمط $integer$ للأس.

ثم قمنا بوضع التابع الأساسي $main$ وبداخله وضعنا العمليات التي سنستخدم التابع على أساسها.

السطر (2): برمجة التابع:

قمنا بإعادة تعريف التابع $double\ pwr(\ double\ x, int\ y)$

حيث $(double\ x, int\ y)$ بارامترات الدخل.

ثم قمنا بتعريف المتغير Res من النمط $double$ وأسندنا له القيمة 1

ثم وضعنا حلقة for بحيث تتكرر الحلقة بعدد الأس الذي أدخله المستخدم

فلوأدخل المستخدم الأس 3 فإن حلقة for ستتكرر 3 مرات.

وضمن حلقة for وضعنا التعليمة $Res = Res * x;$

فمثلاً لو أدخل المستخدم الأساس 2 والأس 4

فإن حلقة for ستتكرر 4 مرات وستكون كالتالي:

$Res = 1 * 2 = 2$
$Res = 2 * 2 = 4$
$Res = 4 * 2 = 8$
$Res = 8 * 2 = 16$

ونلاحظ أن $2^4 = 16$

السطر (3):

قمنا بإعادة القيمة Res أي بعد العمليات يقوم البرنامج بإعادة هذه القيمة فلو لم نعيدها أو وضعنا $return\ 0;$ لضاعت قيمة الـ Res.

سنعيد كتابة **البرنامج** بالطريقة الثانية أي وضع كامل التابع قبل الـ $main$ كما سنستخدم تابع يحسب القوة ولا يعيد قيمة.

```
#include < iostream.h >
void pwr( double x, int y)      (1)
{
double Res = 1;
for(int i = 1; i <= y; i + +)
Res = Res*x;
cout << "Res = " << Res;   (2)
}
void main( )
{
double z; int f;
cout << "Input z, f" << endl;
cin >> z >> f;
pwr (z, f);
}
```

شرح البرنامج

السطر (1):

عرّفنا التابع pwr من النمط $void$ أي لا يعيد قيمة ونلاحظ هنا: بما أننا وضعنا كامل التابع قبل الـ $main$ أننا لم نضع فاصلة منقوطة في نهاية التعريف أما عندما وضعنا الترويسة فقط قبل الـ $main$ وضعنا لها فاصلة منقوطة.

السطر (2) :

نلاحظ وجود تعليمة $cout$ لقيمة الـ Res.. وضعنا هذه التعليمة لأننا لم نعيد قيمة الـ Res أي لم نضع $return\ Res;$.

ثم قمنا بوضع التابع $main$ وبداخله العمليات التي سنستخدم فيها التابع، نلاحظ أننا وضعنا التابع $pwr(z,f)$ دون إسناده للقيمة لأننا لن نحتاج لذلك لأننا نطبع قيمة الـ Res.

تمرين:

اكتب برنامج بلغة $C++$ تقوم من خلاله بحساب قيمة عدد على أن يقوم المستخدم بإدخال أساس هذا العدد وقوته على ان تتم العملية 1000 مرة، وفي كل مرة يمتلك المستخدم الخيار بإدخال الأساس والقوة وذلك باستخدام التوابع.

ملاحظة: قم ببرمجة التابع ولا تستخدم التوابع الجاهزة من ملف $math.h$.

البرنامج:

9

```
#include <iostream.h>
double foo(double,int);      (1)
void main()
{
double x,k;
int y;
for( int i = 1; i <= 1000; i++)     (2)
{
cout << "Input x,y" << endl;
cin >> x >> y;
k = foo(x,y);
cout << "k = " << k << endl;
}
}
```

```
double foo(double k,int l)
{
double Res = 1;
for(int i = 1; i <= l; i++)
Res = Res * x;
return Res;
}
```

شرح البرنامج

السطر (1):

قمنا بتعريف تابع من النمط $double$ وأسميناه foo، ثم أعدنا كتابة كود البرمجة مثل البرامج السابقة:

الفرق الوحيد هو أن النص هنا طلب منا إعادة تكرار العملية 1000 مرة لذلك وضعنا حلقة for

في السطر (2):

تتكرر ألف مرة.

وهناك توابع مخزنة في ذاكرة الحاسوب وحتى نستخدمها يكفي استدعائها باسمها فقط.

مكتبة الدوال الرياضية Math library function

تحتوي مكتبة التوابع الرياضية على العديد من التوابع التي تستخدم في تنفيذ العمليات الرياضية الحسابية.

ملاحظة: عند استعمال التوابع الرياضية في أي برنامج بلغة $C++$ يجب تضمين الملف $math.h$ والذي يحتوي على هذه التوابع.

ويلخص الجدول التالي أهم التوابع الرياضية:

Function	Description	Example
$sqrt(x)$	الجذر التربيعي لـ x	$sqrt(9.0) is\ 3$
$pow(a,b)$	الرفع إلى قوة	a^b $pow(10.5)$
$exp(x)$		e^x
$fabs(x)$	القيمة المطلقة لـ x	$if\ x > 0\ fabs(x) = x$ $= 0\ \ fabs(x) = 0$ $< 0\ \ fabs(x) = -x$
$ceil(x)$	تقريب x لأصغر عدد صحيح أكبر من x	$ceil(9.2) is\ 10.0$ $ceil(-9.8) is -9.0$
$floor(x)$	تقريب x لأكبر عدد صحيح أصغر من x	$floor(9.2) is\ 9$

كما أن هناك بعض التوابع الأخرى نذكر منها:

Function	Description
$sin(x)$	sin الزاوية
$cos(x)$	cos الزاوية
$tan(x)$	tan الزاوية
$a\ tan(x)$	$arctan$ الـ x
$a\ cos(x)$	$arccos$ الـ x
$a\ sin(x)$	$arcsin$ الـ x
$log(x)$	$logarithm$ الـ x
$log\ 10\ (x)$	اللوغاريتم العشري لـ x

العلاقات المنطقية

لدينا في لغة البرمجة العمليات العادية مثل الجمع الطرح والضرب القسمة، ولدينا أيضاً شيء يسمى العلاقات المنطقية وستتحدث عنها في هذه الفقرة:

$$a = (\text{عمليات}) < (\text{عمليات})$$

الإشارة $< >$ تعني أكبر أو أصغر وهي تستخدم للقيام بعملية منطقية، أي أنه إذا تحقق العلاقة تأخذ الـ a قيمة 1 وإذا لم تتحقق تأخذ الـ a قيمة 0

مثال:

$$a = (9 * 2 - 3) > (8\%3 - 2)$$

نلاحظ أنّ:

$$a = 15 > 0$$

والعملية محققة إذاً $a = 1$

$$a == (\text{عمليات}) \&\& (\text{عمليات})$$

الإشارة $\&\&$ تعني And

أي عندما تتحقق العمليتين في ناتج العملية المنطقية هو 1

أما عندما تتحقق واحدة فقط من العمليتين أو لا يتحقق ولا عملية تكون نتيجة العملية المنطقية هي 0

و بالتالي فإن:

$a =$ ناتج العملية المنطقية

مثال:

$$a == (3 + 2 > 4 + 1)\&\&(4 * 3 + 10/2 > 5 * 2 + 3)$$

نلاحظ هنا أن عملية واحدة فقط صحيحة حيث أن:

$$a == (5 > 4\&\&17 > 13)$$

لذلك فإن $a = 0$ لأنه لم تتحقق العمليتين.

$$a = (\text{عمليات}) || (\text{عمليات})$$

الإشارة $||$ تعني (أو) (or)

أي يكفي تحقق عملية واحدة حتى تكون قيمة $a = 1$ أما إذا لم تتحقق ولا عملية تكون قيمة $a = 0$

مثال:

$$a == (3 + 2 > 4 * 1)||(4 * 3 + 10/2 > 5 * 7 + 3)$$

نلاحظ هنا أن عملية واحدة فقط تحققت و هذا يكفي لتكون $a = 1$

إذا كان

$$a! = (\text{عمليات}\&\&\text{عمليات})$$

$$a! = (\text{عمليات})||(\text{عمليات})$$

هنا $a \neq$ ناتج العملية المنطقية

فإذا كان ناتج العملية المنطقية 1 فالنتيجة $a = 0$ و إذا كان ناتج العملية 0 فإن $a = 1$

تعارين محلولة

1. ما هو الملف الرأسي الواجب تضمينه لاستخدام التابع $setw(3)$:

$A. iostream.h$

$B. iomanip.h$ ■

$C. math.h$

$D. stdlib.h$

2. ما هي قيمة المتحول i بعد الحلقة التالية:

$$int\ y\ =\ 0;$$

$$for(int\ i = 0;\ i < 10;\ i++)$$

$$\{\ y += i;\ \}$$

$A.$ غير معرفة

$B. 11$

$C. 10$ ■

$D. 9$

3. ما هو خرج البرنامج التالي:

$$\text{int myList[]} = \{1, 2, 3, 4, 5, 6\};$$

$$\text{for (int i} = 4;\ \text{i} >= 0;\ \text{i}--)$$

$$\text{myList[i} + 1] = \text{myList[i]};$$

$$\text{for (int i} = 0;\ \text{i} < 6;\ \text{i}++)$$

$$\text{cout} << \text{myList[i]} << \text{" "};$$

A. 1 2 3 4 5 6

B. 6 1 2 3 4 5

C. 6 2 3 4 5 1

D. 1 1 2 3 4 5 ■

E. 2 3 4 5 6 1

4. ما هي قيمة sum بعد انتهاء الحلقة التالية:

$$int\ sum\ = 0;\ int\ item = 0;$$

$$do$$

$$\{$$

$$item++;$$

$$sum += item;$$

$$if\ (sum > 4)\ break;$$

$$\}$$

$$while\ (item < 5);$$

$A. 5$

$B. 6$ ■

$C. 7$

$D. 8$

5. أي الكلمات التالية يمكن استخدامه اسماً لمتحول: A. char B. 3small C. mail@scs D. isGoodColor ■	**6. أي من التعابير التالية ينتج القيمة 1 :** $A.\ 2\ \%\ 1$ $B.\ 15\ \%\ 4$ $C.\ 25\ \%\ 5$ $D.\ 37\ \%\ 6$ ■
7. ما هو خرج البرنامج التالي: $void\ f(int\ x = 1, int\ y = 1)$ $\{\ cout\ <<\ x * y\ <<\ endl;\ \}$ $int\ main()\{\ f();\ return\ 0;\ \}$ $A.$ لا شيء $B.\ 1\ \ 0$ $C.\ 1$ ■ $D.\ 0\ \ 1$	**8. ما نتيجة التعليمة التالية :** $cout << ceil(-9.8) << "\ \ " << floor(-9.8) << endl;$ $A.\ -9\ \ -10$ ■ $B.\ 10\ \ \ 9$ $C.\ -9\ \ \ 9$ $D.\ 10\ \ -10$
9. ما هي قيمة التعبير التالي: false && true || true A. true ■ B. false	**10. ما هو ناتج 4 / 45 ؟** $A.\ 10$ $B.\ 11$ ■ $C.\ 11.25$ $D.\ 12$

1. أي من الرموز التالية ليس من رموز اللغة : $A.\ \hat{}$ ■ $B.\ ?:$ $C.\ \%$ $D.\ +=$	**2. ما نتيجة التعليمة التالية :** $cout << ceil(-9.8) << "\ " << floor(9.8) << endl;$ $A. -9\quad -10$ $B. 10\quad 9$ $C. -9\quad 9$ ■ $D. 10\quad -10$
3. ما هي قيمة p بعد تنفيذ الحلقة التالية : $int\ p\ =\ 1;$ $for(int\ k = 2; k >= 0; k--)$ $switch(k)\ \{$ $case\ 0:$ $p -= 2;$ $break;$ $case\ 1:$ $p += 1;$ $break;$ $default: p *= 4;$ $break;\ \}$ $cout << p;$ $A. 2$ $B. 3$ ■ $C. 4$ $D. 5$	**4. ما هو خرج البرنامج التالي :** $\text{int b[4]} = \{2,-1,3,5\};$ $\text{for(int i} = 0;\ \text{i} < 3;\ \text{i}++)$ $\text{if (b[i]} > \text{b[i+1])}$ $\text{b[i+1]} = \text{b[i]} + 1;$ $\text{else b[i]} = \text{b[i+1]} + 1;$ $\text{for(i} = 0;\ \text{i} < 4;\ \text{i}++)$ $\text{cout} << \text{b[i]} << "\ ";$ A. غير ذلك B. 2 3 4 5 C. 3 2 4 6 D. 2 4 6 5 ■

.5 ما هو خرج البرنامج التالي : $int\ Fibo\ (int\ n)$ $\{\ if\ ((n\ ==\ 0)\|\|(n == 1))$ $return\ n;$ $else$ $return\ Fibo(n-1) + Fibo(n-2);\ \}$ $void\ main(\ )$ $\{\ cout\ <<\ Fibo\ (5);\ \}$ $A.3$ $B.4$ $C.5$ ■ $D.6$	**.6 ما هي قيمة المتحول i بعد تنفيذ الحلقة التالية :** $int\ y\ =\ 0, i\ =\ 1;$ $do\{$ $y += i;$ $i++;$ $\}\ while\ (i < 10)$ $A.11$ $B.10$ ■ $C.\ 9$ $D.\ 8$
.7 أي من الكلمات التالية يمكن استخدامه اسماً لمتحول : $A. cout$ $B. 7small$ $C. name@scs$ $D. Float$ ■	**.8 ما هو الملف الرأسي الواجب تضمينه لاستخدام التابع $fabs(x)$:** $A. math.h$ ■ $B. stdlib.h$ $C. iomanip.h$ $D. iostream.h$

1. ما هو الملف الرأسي الواجب تضمينه لاستخدام التابع $rand()$: $A. iostream.h$ $B. iomanip.h$ $C. math.h$ $D. stdlib.h$ ■	2. ما هو نمط المتحول x: $x = sqrt(49);$ $A. int$ $B. float$ $C. double$ ■ $D. char$
3. ما هو خرج الحلقة التالية: $for\ (int\ i = 1;\ i <= 10;\ i++)$ $\{cout \ll i \ll$ " "; $i++;\}$ $A. 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$ $B. 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10$ $C. 1\ 2\ 3\ 4\ 5$ $D. 1\ 3\ 5\ 7\ 9$ ■	4. بفرض $x = 10$ و $y = 10$ ما هي قيمة x بعد تنفيذ التعبير: $(y > 10)\ \&\&\ (x++ > 10)$ $A. 9$ $B. 10$ ■ $C. 11$ $D.$ غير ذلك
5. أي من الكلمات التالية يمكن استخدامه اسماً لمتحول: $A. default$ $B. 2by2$ $C. isGoodName$ ■ $D. mail@scs$	6. أي من التعابير التالية ينتج القيمة 1: $A. 2\ \%\ 1$ $B. 15\ \%\ 4$ $C. 25\ \%\ 5$ $D. 37\ \%\ 6$ ■

7. ما هو خرج البرنامج التالي $int\ RF(int\ n)$:

$\{ if (n <= 1)\ return\ 1;$

$else$

$return\ RF(n-3) + RF(n-1); \}$

$void\ \ main()$

$\{ cout << RF(4); \}$

$A.\, 6$

$B.\, 5$

$C.\, 4$ ■

$D.\, 3$

8. ما هي قيمة المتحول d بعد تنفيذ التعليمات التالية:

$double\ d;$

$int\ a = 5, b = 2, c;$

$c\ =\ (2 * a\ +\ 3 * b)\ \%\ 20;$

$d\ =\ ceil\ (0.1 * c\ /\ b);$

$A.\ \ 1$ ■

$B.\ \ 0$

$C.\ \ -1$

$D.$ غير ذلك

9. إن تعريف مصفوفة أحادية البعد كما يلي صحيح تماماً:

$int\ length = 12;$

$int\ Mat[length];$

$A.$ صح

$B.$ خطأ ■

10. إن تعريف مصفوفة أحادية البعد كما يلي صحيح تماماً:

$const\ int\ length = 12;$

$int\ Mat[length];$

$A.$ صح ■

$B.$ خطأ

1. ما هو الملف الرأسي الواجب تضمينه لاستخدام التابع $floor\ ()$:

$A.\, iostream.h$

$B.\, math.h$ ■

$C.\, stdlib.h$

$D.\, iomanip.h$

2. ما هو نمط المتحول x :

$x\ \ =\ \ 3 * 3/5 * 4;$

$A.\, int$ ■

$B.\, float$

$C.\, double$

$D.\, char$

.3. ما هي نتيجة التعليمة التالية: $int\ a = 6, b = 5, c = 9;$ $cout << c \% a + b / 2;$ $A. 6$ $B. 5$ ■ $C. 5.5$ $D. 4$	.4 بفرض $a = 5, b = 2, c = 10$ ما نتيجة التعبير : $((a + b) \|\| !(--c > 10))$ A. 2 B. 1 ■ C. 0 D. غير ذلك
.5 أي من التعابير التالية لا يمكن استخدامه لزيادة قيمة المتحول $count$ بمقدار 1: $A. count++;$ $B. count = count + 1;$ $C. count += 1;$ $D. count = 1$ ■	.6 إذا كان لدينا x التصريح التالي $char\ x = 'b';$ فإن خرج التعليمة $cout << ++x;$ هو: $A. a$ $B. b$ $C. c$ ■ $D. d$
.7 ما هو خرج البرنامج التالي: $int\ AA(int\ \&n)$ $\{ return\ 3 + n++; \}$ $void\ main()$ $\{int\ x = 1, y = 2;$ $y = AA(x);$ $cout << "x = " << x << ", y = " << y; \}$ $A. x = 2, y = 4$ ■ $B. x = 2, y = 5$	.8 ما هو قيمة المتحول b بعد تنفيذ الحلقة التالية: $int\ a = 2, \quad b = 3;$ $do\{$ $b *= a++;$ $++b;$ $\}\ while(a < 5);$ $A. 87$ $B. 89$ ■

9. من أجل تعريف مصفوفة أحادية البعد عدد عناصرها 5 وإسناد قيمة الصفر لجميع عناصرها نكتب التعليمة : $A. int\ Mat[5];$ $B. int\ Mat[5]\ =\ 0;$ $C. int\ Mat[5] = \{0\};$ ■ $D. int\ Mat[]\ = \{0\};$	10. بفرض لدينا التصريح التالي: $char\ R[] = "lmn";$ما هي قيمة العنصر $R[3]$ ؟ A. l B. m C. n D. غير ذلك ■

اكتب برنامج بلغة $C++$ يقوم بما يلي:

- تصفير عناصر المصفوفة $myArray[5]$
- إدخال 5 أعداد صحيحة وتخزينها في المصفوفة.
- طباعة عناصر المصفوفة المدخلة.

البرنامج:

```
#include < iostream.h >
int main ( ) {
int myArray[5];
for(int i = 0; i < 5, i + +)
myArray [i] = 0;
for(int i = 0; i < 5; i + +)
{
cout << "The value of myArray[" << i << "]: ";
cin >> myArray[i];
}
for(i = 0; i < 5; i + +)
cout << i << ":" << myArray[i] << " ";
cout << endl;
return 0;
}
```

اكتب برنامج بلغة $C++$ يقوم بما يلي :

- ادخال 10 من الاعداد الصحيحة وتخزنها في مصفوفة أحادية البعد $A[10]$.
- يستخدم تابع من النوع int اسمه $search$ للبحث عن عدد مرات تكرار قيمة ما في المصفوفة.

البرنامج:

```
#include < iostream.h >
int search (int [] , int , int );
void main()
{
const int n = 10 ;
int A[n] ;
for (int i = 0 ; i < 10 ; i ++)
{
cout << Eenter number A[ << i << "] = ";
cin >> A[i];
}
int x , sum = 0 ;
cout << "enter the number = ";
cin >> x;
cout << search (A , n , x );
}
int search (int A[] , int n , int x )
{
int sum = 0;
for (int j = 0 ; j < 10 ; j ++)
{
if (A[j] == x)
sum+= 1;
}
return sum ;
}
```

اكتب برنامج بلغة $C++$ يقوم بما يلي :

- ادخال 10 من الاعداد الصحيحة وتخزنها في مصفوفة أحادية البعد $A[10]$.
- يستخدم تابع من النوع $void$ اسمه $sort$ لترتيب الاعداد المدخلة بشكل تصاعدي.
- التابع $main$ يستدعي التابع $sort$ يطبع الاعداد المدخلة بعد عملية الترتيب.

البرنامج:

```
#include <iostream.h>
void sort (int [] , int );
void main()
{
const int n = 10 ;
int A[n];
for (int i = 0 ; i < 10 ; i ++)
{
cout << Enter the number for
 A[ << i << "] = ";
cin >> A[i];
}
sort (A , n );
}
void sort (int A[] , int n)
{
for (int j = 0 ; j < 10 ; j ++)
{
for (int a = 0 ; a < 9 ; a ++)
{
if (A[a] > A[a + 1])
{
int x = A[a];
A[a] = A[a + 1];
A[a + 1] = x;
}
}
}
for (int k = 0 ; k < 10 ; k ++)
{
cout << A[k] << " ";
}
}
```

اكتب برنامج بلغة $C++$ يقوم بما يلي :

- ادخال 20 من الاعداد الصحيحة وتخزنها في مصفوفة أحادية البعد $A[20]$.
- ويحتوي البرنامج على تابعين :
- التابع الأول اسمه $MaxArry()$ مهمته حساب العنصر الأكبر للمصفوفة.
- اما التابع الثاني اسمه $OddArry()$ مهمته حساب مموعة الاعداد الفردية في المصفوفة.
- التابع $main$ يستدعي التابعين و يطبع العنصر الأكبر و مجموع الاعداد الفردية.

البرنامج:

```
#include <iostream.h>
int MaxArray ( int [] , int );
int OddArray ( int [] , int );
void main()
{
const int n = 20 ;
int A[n] ;
for (int i = 0 ; i < n ; i ++)
{
cout << Enter the number of A"[ << i << ]=" << endl;
cin >> A[i];
}
cout << MaxArray ( A , n ) << endl;
cout << OddArray ( A , n ) << endl;
}
int MaxArray (int A[] , int n )
{
int max = A[0] ;
for (int j = 0; j < n; j ++)
{
if (A[j] > max)
max = A[j];
}
return max ;
}
int OddArray ( int A[] , int n )
{
int sum = 0;
for (int k = 0 ; k < n ; k ++)
{
if (A[k]%2! = 0)
sum+= A[k];
}
return sum ;
}
```

اكتب برنامج بلغة $C++$ يقوم بما يلي :

- ادخال 20 من الاعداد الصحيحة وتخزنها في مصفوفة أحادية البعد $A[20]$.
- ويحتوي البرنامج على تابعين :
- التابع الأول اسمه $SumAve()$ مهمته حساب المتوسط الحسابي لعناصر المصفوفة.
- اما التابع الثاني اسمه $EvenArray()$ مهمته حساب مجموع الاعداد الزوجية لعناصر المصفوفة.
- التابع $main$ يستدعي التابعين و يطبع المتوسط الحسابي ومجموع الاعداد الزوجية.

البرنامج:

```
#include <iostream.h>
int SumAve(int[] , int);
int EvenArray(int[], int);
void main() {
const int n = 5 ;
int A[n];
for(int i = 0; i < n; i + +)
{cout << A[ << i << "] = ";
cin >> A[i]; }
SumAve(A , n);
cout << the Avg= << SumAve(A , n)
        << endl;
cout << the Even Array << EvenArray(A, n)
        << endl; }
int SumAve(int A[] , int n)
{int sum = 0 ; double Avg ;
for (int j = 0 ; j < n ; j + +)
{sum+= A[j]; }
Avg = sum/n;
return Avg ; }
int EvenArray(int A[], int n)
{
int Even = 0 ;
for(int k = 0 ; k < n ; k + +)
{
if (A[k]%2 == 0)
Even+= A[k];
}
return Even;
```

انتهى الملخص ... نرجو أن تكونوا قد استفدتم ...